

A new approach to present prototypes in clustering of time series

Saeed R. Aghabozorgi, Teh Y. Wah, Amineh Amini, and Mahmud R. Saybani

Abstract—There are considerable advances in clustering time series data in data mining concept. However, most of which use traditional approaches and try to customize the algorithms to be compatible with time series data. One of the significant problems with traditional clustering is defining prototype specially in partitional clustering where it needs centroids as representative of each cluster. In this paper we present a novel effective approach to define the prototypes based on time series nature. The prototype is constructed based on fuzzy concept efficiently. Moreover, it is demonstrated how the prototypes are moved in iterations. We will present the benefits of the proposed prototype by implementing a real application: Customer transactions clustering.

I. INTRODUCTION

THERE are different approaches to analyze time series data, which clustering is one of the most frequently used techniques [1], owing to its exploratory nature, and its application as a pre-processing phase in more complex data mining algorithms. There are variety of studies, projects and surveys that have noted different approaches and comparative respects of time series clustering [2-13]. Clustering of time series data has three overall problems which do not exist in traditional clustering algorithm (static objects clustering): representation method, distance measurement and clustering algorithm. Choosing a proper approach to represent time series data as a low dimension data is the problem statement of many papers. Another respect of time series clustering is finding an adequate distance measurement between time series data, whether between raw time series or dimensionality reduced time series. Finally, choosing an accurate and fast clustering algorithm, compatible with time series data is a challenge for some researches.

In order to compare time series with irregular sampling intervals and length, it is of great significance to adequately determine the similarity of time series. There is different distance measurements designed for specifying similarity between time series. The Hausdorff distance and modified Hausdorff (MODH), Euclidean distance, HMM-based distance, dynamic time warping (DTW), Euclidean distance in a PCA subspace, and longest common subsequence

(LCSS) are the most popular distance measurement methods used for time series data. Zhang et al. [14] has performed a complete survey on the aforementioned distance measurements comparing them in different applications.

Dynamic time warping (DTW) [15, 16] is one of the most famous algorithms for measuring similarity between two sequences with irregular-lengths without any discretization which can be different in terms of time or speed. In DTW method, the sequences are "warped" non-linearly in the time dimension to determine a measure of their similarity independent of certain non-linear variations in the time dimension. It makes two pairs of the nearest points from each sequence, allowing a one-to-many matching. However, the comparison in DTW does not perform a structural comparison of the time series because the comparison is based on the local dissimilarity. Another roughly similar measurement is LCSS (Longest Common Sub-Sequence) which is useful especially for unequal length data, and it is more robust to noise and outliers than DTW because all points do not need to be matched. In LCSS a point with no good match can be ignored to prevent unfair biasing. For aforementioned reasons LCSS is employed as distance measure for our methodology. However, defined prototypes for clusters are not based upon the measurement, whether DWT or LCSS is used as measurement. That is, if we use DWT or LCSS special measurements, it is not proper to utilize the average value (centroid) or median [17] as prototype of the cluster, because these kind of prototypes are based on Euclidean space. In this situation it is more accurate if a prototype is defined based on each used measurement. For example, in figure 1, the centroid (mean) is constructed based on the mean values of two time series. The results show that surprisingly, the final clusters using this prototype are not accurate enough as we show in experimental results.

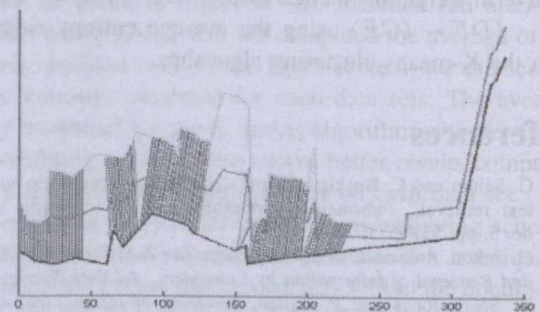


Fig. 1. Centroid (mean) prototype between two time series

S. R. Aghabozorgi is with University of Malaya, Department of Information Science, Faculty of Computer Science & Information Technology Building, University of Malaya, 50603 Kuala Lumpur, Malaysia (e-mail: saeed@siswa.um.edu.my).

Y. W. Teh The (e-mail: tehyw@um.edu.my), A. Amini (e-mail: amineh@siswa.um.edu.my) and M. R. Saybani (e-mail: saybani@siswa.um.edu.my) are with University of Malaya, Department of Information Science, Faculty of Computer Science & Information Technology Building, University of Malaya, 50603 Kuala Lumpur, Malaysia

Another approach is using mean value of match points when LCSS is used as measurement. Using this method also cannot solve the issue because the length of prototype is

decreased when there are many time series belong to a cluster (Figure 2).

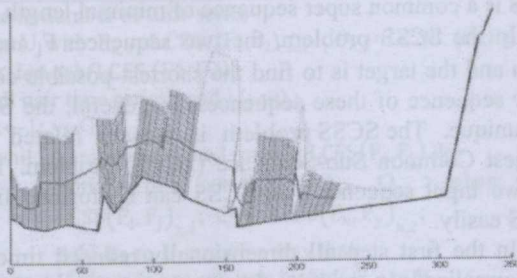


Fig. 2. Prototype based on match points

One of the recent affords in this area is [18] where authors formulated the prototype computation problem as a optimization task, and proposed an local search solution to solve it. They claim that this approach improves the k -medoids approach.

In this work, we present an optimal prototype based on LCSS measurement and then we show how using this prototype raises the accuracy of final result rather than using average or median approach.

The rest of this paper is organized as follows. In section 2, the terminology is described, and then in section 3 the methodology is presented. The algorithm is applied on real time series data sets and the experimental results are reported in sections 4. In section 5 the results are evaluated, and in section 6 conclusions and achievements are drawn.

II. TERMINOLOGY

We start by providing some basic notation and preliminary definitions.

Definition 1. Time series: A time series $F_i = \{f_1, \dots, f_t, \dots, f_T\}$ is a ordered set of flow vectors which indicate the spatiotemporal characteristics of moving objects at any time t of the total track life T [19]. A flow vector or feature vector $f_t = [X, Y, Z, \dots]$ generally represents location and dynamics in the domain. However, we limit ourselves to just a spatial location $f_t = [X]$ in this work for the sake of simplicity. We assume $M = \{F_1, \dots, F_i, \dots, F_n\}$ is a collection of time series in a domain, where F_i represents i -th time series ($i = 1, \dots, n$) in the domain.

Definition 2. Similar time series: Two time series F_i and F_j are defined as similar if and only if $D(F_i, F_j) < \epsilon$, where $D(F_i, F_j)$ is a function or process for calculating similarity between F_i and F_j , and ϵ is a specified threshold value [20].

A. Longest common Sub-sequence

In this section we shortly explain the LCSS employed as distance measure for our methodology.

Definition 3. Longest Common Sub-Sequence: Given F_i as a time series and f_t as feature vector at time t in time series F_i , if f_{qt} is the feature q -th of time series for $q = \{1, \dots, p\}$ at

time t and if p is number of features describing each object, then the LCSS distance is defined as [21]:

$$LCSS(F_i, F_j) = \begin{cases} 0, & T_i = 0 \vee T_j = 0 \\ 1 + LCSS(F_i^{T_i-1}, F_j^{T_j-1}), & d_E(f_i, T_i, f_j, T_j) < \epsilon \text{ and } |T_i - T_j| < \delta \\ \max(LCSS(F_i^{T_i-1}, F_j^{T_j}), LCSS(F_i^{T_i}, F_j^{T_j-1})), & \text{otherwise} \end{cases} \quad (1)$$

Where the $LCSS(F_i, F_j)$ value states the number of matching points between two time series and $F_i = \{f_1, \dots, f_t\}$ specifies all the flow vectors in time series F_i up to time t . Additionally, in this formula, δ is an integer value which constricts the length of the warping and $0 < \epsilon < 1$ is a real number as the spatial matching threshold to cover elements with real values. More precisely, ϵ is a tolerance threshold to find the set of flow vectors in a time series that are within distance ϵ from a point (flow vector) in another time series. The LCSS also has the ability of computing efficiently using dynamic programming like similar to what has been done with DTW.

In this paper, a customized distance measure is defined based on LCSS as:

$$D_{LCSS}(F_i, F_j) = 1 - \frac{LCSS(F_i, F_j)}{\text{mean}(T_i, T_j)} \quad (2)$$

Where, using $\text{mean}(T_i, T_j)$ instead of $\min(T_i, T_j)$ results in taking the length of both time series into account.

B. Fuzzy C-Means (FCM) algorithm

One of the most extensively used clustering algorithms is the Fuzzy C-Means (FCM) algorithm presented by Bezdek [22]. FCM works by partitioning a collection of n vectors into c fuzzy groups and finds a cluster center in each group such that the cost function of dissimilarity measure is minimized. Bezdek introduced the idea of a "fuzzification parameter" (m) in the range $[1, n]$ which determines the degree of fuzziness (weighted coefficient) in the clusters. Essentially, the parameter m controls the permeability of the cluster horizon which can be viewed as an n -dimensional cloud moving out from a cluster center [23].

Given c as number of classes, v_j , centre of class j for $j = \{1, \dots, c\}$, n as the number of time series and μ_{ij} as the degree of membership of the time series i to cluster j for $i = \{1, \dots, n\}$, distance of each time series F_i from each cluster center(v_j) is denoted as such:

$$d_{ij} = (F_i, v_j) = D_{LCSS}(F_i, v_j) \quad (3)$$

Let the centers be shown by $v_j = \{v_1, \dots, v_c\}$ and each time series by F_i that $i = \{1, \dots, n\}$ and d_{ij} as distances between centers and time series. Therefore, the membership values μ_{ij} are obtained with:

$$\mu_j(x_i) = \frac{\left(\frac{1}{d_{ji}}\right)^{\frac{1}{m-1}}}{\sum_{k=1}^p \left(\frac{1}{d_{ki}}\right)^{\frac{1}{m-1}}} \quad (4)$$

And the sum of cluster memberships for a time series equals 1:

$$\sum_{j=1}^c \mu_{ij}(F_i) = 1, \forall i \in \{1, \dots, n\} \quad (5)$$

The FCM objective function (standard loss) that is attempted to be minimized takes the form:

$$J = \sum_{j=1}^c J_j = \sum_{j=1}^c \sum_{i=1}^n [\mu_{ij}]^m d_{ij} \quad (6)$$

where μ_{ij} is a numerical value between $[0; 1]$; d_{ij} is the Euclidian distance between the j th prototype and the i th time series; and m is the exponential weight which influences the degree of fuzziness of the membership matrix.

In different iterations, the membership values of the time series are calculated, and then the prototypes (cluster centers) are recomputed. In order to update a new cluster center value, the following formula is employed:

$$v_j = \frac{\sum_{i=1}^n (\mu_{ij})^m (F_i)}{\sum_{i=1}^n (\mu_{ij})^m} \quad \forall j \in \{1, \dots, c\} \quad (7)$$

III. PROTOTYPE CALCULATION BASED ON EXISTING TIME SERIES INSIDE A CLUSTERS

In this methodology each time series is assigned to a cluster, whose prototype (centroid) is the nearest. The prototype of a cluster has to be constructed in such a way that:

1. The prototype has to be changed based on changes of time series inside the cluster
2. Time series inside a cluster should have most similarity to their cluster's prototype

This problem is break down in two sub-problems:

- A. Making a centre for a group of existing time series inside a cluster
- B. Moving existing centers based on time series inside the clusters

A. Defining a prototype for a group of time series

In order to construct a prototype based on exist time series in a cluster, a new time series is defined as prototype of each cluster. In this step, we use the Shortest Common Super Sequence (SCSS) to make the prototype.

Definition 4. Shortest Common Super Sequence: Given two sequences $F_i = \langle f_{i1}, \dots, f_{it}, \dots, f_{im} \rangle$ and $F_j = \langle f_{j1}, \dots, f_{jt}, \dots, f_{jn} \rangle$, a sequence $U_k = \langle$

$u_{k1}, \dots, u_{kt}, \dots, u_{km} \rangle$ is a common super sequence of F_i and F_j , if U_k is a super sequence of both F_i and F_j . The SCSS is a common super sequence of minimal length.

In the SCSS problem, the two sequences F_i and F_j are given and the target is to find the shortest possible common super sequence of these sequences. In general, the SCSS is not unique. The SCSS problem is closely related to the Longest Common Sub-Sequence (LCSS) problem. That is, for two input sequences, an SCSS can be formed from an LCSS easily.

In the first step all dimensionally reduced time series are normalized. In the next figures three original (Figure 3) and normalized (Figure 4) time series inside a cluster are depicted.

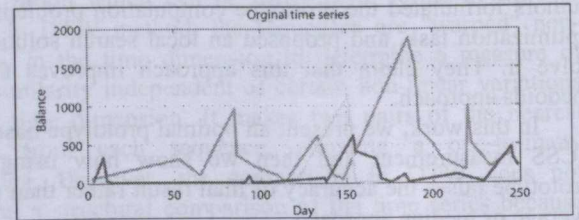


Fig. 3. Raw time series before normalization

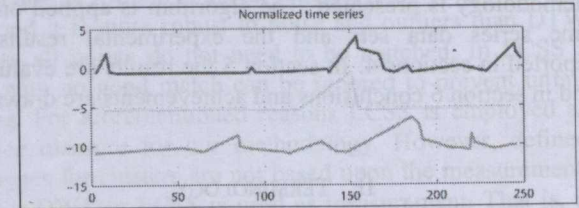


Fig. 4. Normalized time series

In order to find SCSS among time series, the LCSS among the time series are used. The following pictures illustrate the match points (Figure 5) among two time series calculated by dynamic programming.

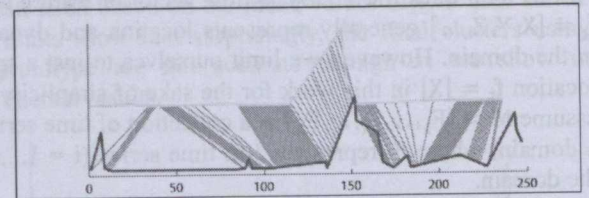


Fig. 5. Match points between time series based on LCSS

In order to make the prototype based on the clusters' members, a novel algorithm is presented in table 1. In this algorithm, cluster C is denoted as $C = \{F_1, \dots, F_i, \dots, F_n\}$ and F_i as a member of cluster C . For each $F_i = \{f_{i1}, \dots, f_{it}, \dots, f_{ip}\}$, f_{it} is a point of time series F_i and $LCSS(F_i, F_j)_{x+2}$ is the longest common sub sequence of time series F_i and F_j and x is the total number of common points between them.

TABLE 1
PSEUDO CODE FOR PROTOTYPE CALCULATION BASED ON THE LONGEST
COMMON SUB SEQUENCE.

```

Initialize n=number of time series
Initialize U with  $P_{k,1}=LCSS(F_1, F_2)_{k,1}$ ,  $P_{k,2}=LCSS(F_1, F_2)_{k,2}$ ,
 $1 < k < length(LCSS(F_1, F_2))$ 
for all pair time series i and j ( $i < j$ ):
    x←1; y←1;
    while there are unvisited rows in  $LCSS(F_i, F_j)$  do
        initial a new pair  $Q_x = \langle Q_{x,i}, Q_{x,j} \rangle$  where  $Q_{x,i} =$ 
 $LCSS(F_i, F_j)_{x,1}$ ;  $Q_{x,j} = LCSS(F_j, F_i)_{x,2}$ ;
        Add  $Q_x$  to U in such a way that:
        If there is not any pair include  $P_{y,i}$  in  $P_y$ 
            increase y;
        elseif  $Q_{x,i} < P_{y,i}$  then
            insert  $Q_x$  before  $P_x$  in U; increase x,y;
        elseif  $Q_{x,i} = P_{y,i}$  then
             $P_{y,j} = Q_{x,j}$ ;
            increase x,y;
        elseif  $Q_{x,i} > P_{y,i}$  then
            increase y;
        end if
    end while
end for
for each  $P_j$  in U
     $V_j = mean(f_{p_{j,x}}, f_{p_{j,y}}) \forall x, y \text{ which } \langle p_{j,x}, p_{j,y} \rangle \in P_j$ 
end for
return  $V_j$  as prototype of cluster O

```

In this algorithm, we define an ordered set $U = \langle P_1, \dots, P_k, \dots \rangle$ for showing the match points' indices which construct the prototype. The P_k is defined as a none-ordered set as $P_k = \langle (p_{k,1}, p_{k,2}), \dots, (p_{k,i}, p_{k,j}), \dots \rangle$ which includes a set of pair points of time series i and j which construct the k-th point of prototype.

E.g. $P_3 = \langle (i2, j2)(i2, k4)(j2, k2) \rangle$ denotes that P_3 is made from second points of time series i, j, k, and the fourth point of k.

At first a common super sequence is made based on the LCSS of each pair of cluster members. Then intermediate points between each pair of time series are considered in order to turn the common time series into a SCSS. This time series is denoted as the SCSS of all match points among the time series in the same cluster. Based on this definition, the prototype can be regarded as the shortest sequence that includes all LCSSs among the time series within the cluster. Then the prototype of cluster j is defined as:

$$V_j = SCSS(F_1, \dots, F_i, \dots, F_z), \quad \forall i \in \{\text{membership of cluster } j\}, \forall j \in \{1, \dots, c\} \quad (7)$$

and the value of each point of V_j is calculated by the average of the value of each common pair points in the SCSS. For the sake of simplicity, an example of a prototype calculated for a cluster with three sample time series is provided in table 2.

TABLE 2
EXAMPLE OF CALCULATING OF PROTOTYPE OF A CLUSTER BASED ON THE
LONGEST COMMON SUB SEQUENCE

If we consider F_i, F_j and F_k as example time series within a cluster,
 $F_i = \langle f_{i,1}, \dots, f_{i,7} \rangle = \langle 2.1, 3.2, 1.1, 2.7, 4.8, 2.9, 1.5 \rangle$
 $F_j = \langle f_{j,1}, \dots, f_{j,5} \rangle = \langle 2.8, 3.3, 2.8, 2.2, 1.4 \rangle$
 $F_k = \langle f_{k,1}, \dots, f_{k,9} \rangle = \langle 2.7, 3.2, 2.2, 3.2, 2.8, 2.8, 2.3, 1.5, 1.6 \rangle$
 and, if LCSS of each pairs of the time series is assumed as follow:

$$LCSS(F_i, F_j) = \begin{bmatrix} i2 & j2 \\ i4 & j3 \\ i7 & j5 \end{bmatrix}, \quad LCSS(F_i, F_k) = \begin{bmatrix} i1 & k3 \\ i2 & k4 \\ i6 & k5 \\ i7 & k9 \end{bmatrix},$$

$$LCSS(F_j, F_k) = \begin{bmatrix} j1 & k1 \\ j2 & k2 \\ j3 & k5 \\ j4 & k7 \\ j5 & k8 \end{bmatrix}$$

Then, the prototype indexes is calculated in three steps as:

(1*) the $U_{j,n}$ is defined based on first common piecewise (F_i) between $LCSS(F_i, F_j)$ and $LCSS(F_i, F_k)$

(2*) Updating based on second common piecewise (F_j) between $LCSS(F_i, F_j)$ and $LCSS(F_j, F_k)$

$$(1^*) = \begin{bmatrix} i1 & - & k3 \\ i2 & j2 & k4 \\ i4 & j3 & - \\ i6 & - & k5 \\ i7 & j5 & k9 \end{bmatrix} \rightarrow (2^*) = \begin{bmatrix} - & j1 & k1 \\ i2 & j2 & k4, k2 \\ i4 & j3 & k5 \\ i6 & - & k5 \\ - & j4 & k7 \\ i7 & j5 & k9, k8 \end{bmatrix}$$

(3*) common piecewise among F_i, F_j and F_k

$$(3^*) = SCSS(F_i, F_j, F_k) = \begin{bmatrix} (i1, k3) \\ (j1, k1) \\ (i2, j2)(i2, k4)(j2, k2) \\ (i4, j3)(j3, k5) \\ (i6, k5) \\ (j4, k7) \\ (i7, j5)(i7, k9)(j5, k8) \end{bmatrix}$$

(4*) Value of prototype based on common piecewise among F_i, F_j and F_k :

$$(4^*) = V_j = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \end{bmatrix} = \begin{bmatrix} mean(f_{i1}, f_{k3}) \\ mean(f_{j1}, f_{k1}) \\ mean(f_{i2}, f_{i2}, f_{j2}, f_{j2}, f_{k4}, f_{k2}) \\ mean(f_{i4}, f_{j3}, f_{j3}, f_{k5}) \\ mean(f_{i6}, f_{k5}) \\ mean(f_{j4}, f_{k7}) \\ mean(f_{i7}, f_{i7}, f_{j5}, f_{j5}, f_{k9}, f_{k8}) \end{bmatrix}$$

$$= \begin{bmatrix} \frac{2.1 + 2.2}{2} \\ \frac{2.8 + 2.7}{2} \\ \frac{3.2 + 3.3 + 3.2 + 3.2 + 3.3 + 3.2}{6} \\ \frac{2.7 + 2.8 + 2.8 + 2.8}{4} \\ \frac{2.9 + 2.8}{2} \\ \frac{2.2 + 2.3}{2} \\ \frac{1.5 + 1.4 + 1.5 + 1.6 + 1.4 + 1.5}{4} \end{bmatrix} = \begin{bmatrix} 2.15 \\ 2.75 \\ 3.23 \\ 2.77 \\ 2.85 \\ 2.25 \\ 1.48 \end{bmatrix}$$

The figure 4 illustrates calculated prototype for a cluster include three time series presented in figure 3.

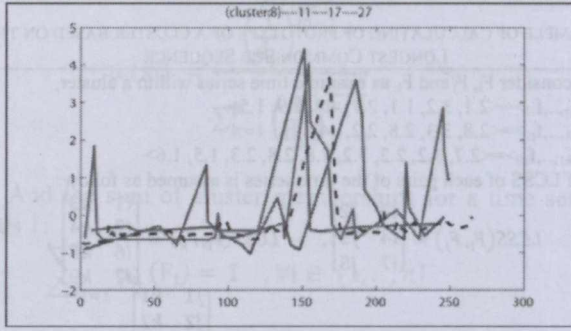


Fig. 5. The calculated prototype of normalized time series

B. Moving prototypes

In order to move the prototypes, the prototypes to be updated according to the membership of all objects. Similarly, the fuzziness of the time series is utilized in order to update prototypes in this approach. In table 3, a new algorithm is presented to explain the details of updating prototypes. In this algorithm, $F_i = \{f_{i1}, \dots, f_{it}, \dots, f_{ip}\}$ is a time series with length p , and f_{it} is a point of the time series F_i at time t , and matrix $LCSS(F_i, V_j)_{r \times 2}$ indicates match points (LCSS) of time series F_i and prototype V_j , where $LCSS(F_i, V_j)$ has a dimension of $r \times 2$ (r is the number of match points)

We define an ordered set $U = \langle P_1, \dots, P_k \rangle$ correspond to V_j as prototype. The P_k is defined as a none-ordered set as $P_k = \langle p_{k,1}, p_{k,2}, \dots, p_{k,b} \rangle$, $b < n$ which includes a set of index of points of time series (which have an specific condition explained further). E.g. $P_3 = \langle i7, y3, z6 \rangle$ that indicates the f_{i7} point of time series F_i and f_{y3} point of time series F_y and so on.

TABLE 3
PSEUDO CODE TO UPDATE PROTOTYPE BASED ON TIME SERIES INSIDE A CLUSTER

```

initialize  $\mu_{\min} = \frac{1}{c}$  (determine a threshold equal to the inverse of the class
number for  $c$  as number of clusters)
initialize  $\mu_{\text{mean},j} = \frac{\sum_{i=1}^z [\mu_{ij}]}{z} \quad \forall i \in \{\text{members of cluster } j\}, z =$ 
number of members,
initialize a set  $U$ , length=length( $V_j$ )
for each time series  $F_i$  with  $\mu_{ij} > \mu_{\min}$ :
    for each pair of match point in  $LCSS(F_i, V_j)$ ,  $r$ : the row number in matrix
     $LCSS(F_i, V_j)$ 
        initialize  $P_{LCSS(F_i, V_j)_{r,1},i} = LCSS(F_i, V_j)_{r,2}$ ;
    end for
end for
/// extending the prototype
for every time series  $F_i$  that  $\mu_{ij} > \mu_{\text{mean}}$ :
     $h \leftarrow 1$ ;  $r \leftarrow 1$ 
    while  $r \leq \text{length}(F_i)$ 
        initialize a new pair  $Q$  as  $Q = \langle f_{it}, \mu_{ij} \rangle$ ;
        initialize set  $Q_x = \langle Q_{x,1}, \dots, Q_{x,n} \rangle$   $Q_{x,n} = r$ ;
        read  $P_h$  from  $U$ 
        if  $r < P_{h,i}$ 
            insert  $P$  before  $P_h$  in  $U$ ;
            increase  $h, r$ ;
        elseif  $r = P_{h,i}$ 
            increase  $h, r$ ;

```

```

elseif  $r > U_{hi}$ 
    increment  $h$ ;
end
end
for each  $P_h$  in  $v$ 
 $V'_{jt} = \frac{\sum_{i=1}^b (\mu_{ij})^m (f_{ix})}{\sum_{i=1}^b (\mu_{ij})^m}$ ,  $b = \text{length}(P_h)$ 
end if

```

In the mentioned algorithm above, a threshold μ_{\min} is defined and then prototypes are updated based on the time series with memberships more than μ_{\min} (candidate time series). This threshold is required in order to ignore the noise by omitting time series with a lower fuzziness value. Additionally, it prevents prototypes from stretching incrementally over time. μ_{\min} equals to the inverse of the class number value:

$$\mu_{\min} = \frac{1}{c} \quad (8)$$

In accordance with the definition, only a specific part of time series (candidate time series) is considered in the calculation of prototypes. For each candidate time series, corresponding points must be found, that is, match points (LCSS) between time series and prototypes. The set U of the prototype is initialized according to following equation:

$$P_{LCSS(F_i, V_j)_{r,1},i} = LCSS(F_i, V_j)_{r,2} \quad (9)$$

where $P_{LCSS(F_i, V_j)_{r,1},i}$ indicates the index of common points (match points) between all of the time series assigned to a cluster with an acceptable membership.

Until now, only the common points among all candidate time series have been considered. In the next step, the U set is updated based on some parts of candidate time series which, although they do not have match points with the prototype, they have higher memberships than $\mu_{\text{mean},j}$ ($\mu_{ij} > \mu_{\text{mean},j}$). These points are inserted between common points of U in order to take only sub-sequences that have acceptable membership into account. For cluster j with z members, $\mu_{\text{mean},j}$ is calculated as:

$$\mu_{\text{mean},j} = \frac{\sum_{i=1}^z [\mu_{ij}]}{z} \quad \forall i \in \{\text{members of cluster } j\}, \quad (10)$$

$z = \text{number of members of cluster } j$

In corresponding with matrix U , matrix S stores the value of each point f_{it} and its membership μ_{ij} . That is, for each record of U (each point of main centre), there are a set of point values and their membership's value in S . Let t be one of these points in U . Now if point t of the new center is matched with h different points with value X_i and membership μ_{ij} from h different time series, point t of the updated prototype V'_j is shown as:

$$V_{jt} = \frac{\sum_{i=1}^b (\mu_{ij})^m (f_{ix})}{\sum_{i=1}^h (\mu_{ij})^m} \quad \forall i \in \{1, \dots, h\}, \quad (11)$$

$$\forall t \in \{1, \dots, n\}$$

Table 4 shows an example of selecting candidate time series among four time series and then updating prototypes based on $\mu_{mean,j}$

TABLE 4
EXAMPLE OF UPDATING PROTOTYPE BASED ON EXISTING TIME SERIES IN A CLUSTER

Given four assumed time series, candidate time series are declared for cluster j as:

Time series:	μ_{min}	$\mu_{mean,j}$
$F_i = \langle f_{i1}, \dots, f_{i7} \rangle$	$\mu_{i,j} > \mu_{min}$	$\mu_{i,j} > \mu_{mean,j}$
$F_y = \langle f_{y1}, \dots, f_{y5} \rangle$	$\mu_{y,j} > \mu_{min}$	$\mu_{y,j} > \mu_{mean,j}$
$F_x = \langle f_{x1}, \dots, f_{x7} \rangle$	$\mu_{x,j} < \mu_{min}$	$\mu_{x,j} < \mu_{mean,j}$
$F_z = \langle f_{z1}, \dots, f_{z9} \rangle$	$\mu_{z,j} > \mu_{min}$	$\mu_{z,j} < \mu_{mean,j}$

The LCSS between each pairs of time series are assumed as :

$$LCSS(F_i, V_j) = \begin{bmatrix} i2 & v2 \\ i4 & v3 \\ i7 & v5 \end{bmatrix}, \quad LCSS(F_y, V_j) = \begin{bmatrix} y1 & v1 \\ y2 & v2 \\ y3 & v5 \\ y4 & v7 \end{bmatrix}$$

$$LCSS(F_z, V_j) = \begin{bmatrix} z1 & v3 \\ z2 & v4 \\ z6 & v5 \end{bmatrix}, \quad LCSS(F_x, V_j) = \begin{bmatrix} x1 & v1 \\ x3 & v2 \\ x7 & v5 \\ x8 & v6 \end{bmatrix}$$

(1*) step 1: Find common points between V and time series with membership more than μ_{min} , in this case: (F_i, F_y, F_z)

(2*) step 2: Extend the matrix U with some parts of time series that has membership more than $\mu_{mean,j}$ (F_i, F_y)

$$V_j = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ \vdots \\ v_n \end{bmatrix} \quad U = (1^*) \begin{bmatrix} - & - & - \\ i2 & y2 & - \\ - & - & - \\ i4 & - & z1 \\ - & - & z2 \\ - & - & - \\ - & - & - \\ - & - & - \\ i7 & y3 & z6 \\ - & y4 & - \\ \vdots & \vdots & \vdots \end{bmatrix} \rightarrow (2^*) \begin{bmatrix} i1 & - & - \\ - & y1 & - \\ i2 & y2 & - \\ i3 & - & - \\ i4 & - & z1 \\ - & - & z2 \\ i5 & - & - \\ i6 & - & - \\ - & - & z3 \\ - & - & z4 \\ i7 & y3 & z6 \\ - & y4 & - \\ \vdots & \vdots & \vdots \end{bmatrix}$$

$$\rightarrow \begin{bmatrix} i1 & y1 \\ i2, y2 \\ i3 \\ i4, z1 \\ z2 \\ i5 \\ i6 \\ z3 \\ z4 \\ i7, y3, z6 \\ y4 \\ \vdots \end{bmatrix} \begin{bmatrix} \mu_{i,j} \\ \mu_{y,j} \\ \mu_{i,j}, \mu_{y,j} \\ \mu_{i,j} \\ \mu_{i,j}, \mu_{z,j} \\ \mu_{i,j} \\ \mu_{i,j} \\ \mu_{z,j} \\ \mu_{z,j} \\ \mu_{y,j}, \mu_{z,j}, \mu_{i,j} \\ \mu_{y,j} \\ \mu_{z,j} \end{bmatrix} \rightarrow V'_j = \begin{bmatrix} v'_1 \\ v'_2 \\ v'_3 \\ v'_4 \\ v'_5 \\ \vdots \\ v'_n \end{bmatrix}$$

IV. EXPERIMENTAL RESULTS

In this paper, we focus on segmentation of bank customers as identifiable objects to show how we utilize the presented method for clustering of customers.

For a bank in Malaysia, similar customers based on their daily transactions are desirable. The profile of similar

customer is used for decision making, fraud detection, campaigns, etc. In order to find accurate similar transactions on all accounts, we try to cluster cards based on their balance on each day. We have applied fuzzy clustering algorithm on different cardinality of dataset of the customer time series databases, but with the proposed prototype approach.

Our dataset is a collection of time series which includes 365 days of outstanding amount of 10,000 credit cards. Each time series in this dataset is presented by 200 to 365 observations.

V. EVALUATION

We call our prototype calculation approach FPT (Fuzzy Prototype of Time series), and compare its accuracy with different clustering algorithms.

clustering of time series is an unsupervised process and there are no predefined classes and no examples that can show that the clusters found by the clustering algorithms are valid [24]. Therefore, it is necessary to use some validity criteria. In order to prove that our approach is more efficient than utilized conventional prototypes, we employed the FCM algorithms with different prototypes (median, mean and FPT) for comparison purpose. We have collected different amount of records from our dataset to show how it works in terms of accuracy.

In order to evaluate clusters in terms of accuracy, we used Squared Error (SSE), the most common measure. For each time series, the error is the distance to the nearest cluster. To get SSE, we used following formula:

$$SSE = \sum_{j=1}^c \sum_{F_i \in C_j} (\text{dist}(F_i, V_j))^2 \quad (12)$$

Where, F_i is a time series in cluster C_j and V_j is the prototype for cluster C_j .

The results illustrated in figure 10 and 11 are related to average of SSE for 10 times run of the FCM algorithms with three different approaches. The results show that the presented algorithm is competitive with other algorithms in terms of accuracy.

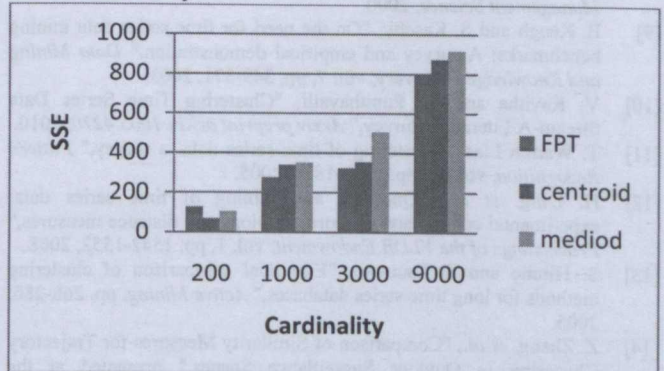


Fig. 6. Accuracy of using FPT in clustering crosses different cardinalities.

VI. CONCLUSION

The purpose of the current study was to present a method to present a prototype for time series clusters efficiently. We developed a novel method for constructing prototypes based on its ability to be accurate enough. The defined prototype can be updated based on fuzzy concept through iterations as well. We discussed about this fact that if the prototype for partitioning algorithms is computed precisely, the clusters is more accurate.

In order to show experimental results, PFT methodology was implemented on time series data of a bank to perform the segmentation. Moreover, we applied two more frequently used prototypes in clustering algorithms on our dataset to compare them with the developed approach (FPT) in terms of accuracy. The results of this study indicate that this method is much more efficient than conventional prototypes used in clustering algorithms. It is because of considering the common points of time series in clusters instead of whole data points.

However, further research needs to be done in order to evaluate FPT in terms of execution time and accuracy of data clusters in different datasets with different dimensions to understand its potentials and limitations.

VII. REFERENCES

- [1] M. Halkidi, *et al.*, "On clustering validation techniques," *Journal of Intelligent Information Systems*, vol. 17, pp. 107-145, 2001.
- [2] P. Cotofrei and K. Stoffel, "Classification rules+ time= temporal rules," *Computational Science—ICCS 2002*, pp. 572-581, 2002.
- [3] G. Das, *et al.*, "Rule discovery from time series," *Knowledge Discovery and Data Mining*, pp. 16-22, 1998.
- [4] T. Fu, *et al.*, "Pattern discovery from stock time series using self-organizing maps," 2001, pp. 26-29.
- [5] M. Gavrilov, *et al.*, "Mining the stock market: Which measure is best," 2000.
- [6] X. Jin, *et al.*, "Indexing and mining of the local patterns in sequence database," *Intelligent Data Engineering and Automated Learning—IDEAL 2002*, pp. 39-52, 2002.
- [7] E. Keogh and C. Ratanamahatana, "Exact indexing of dynamic time warping," *Knowledge and Information Systems*, vol. 7, pp. 358-386, 2005.
- [8] P. Tino, *et al.*, "Temporal pattern recognition in noisy non-stationary time series based on quantization into symbolic streams: Lessons learned from financial volatility trading," *Report Series for Adaptive Information Systems and Management in Economics and Management Science*, 2000.
- [9] E. Keogh and S. Kasetty, "On the need for time series data mining benchmarks: A survey and empirical demonstration," *Data Mining and Knowledge Discovery*, vol. 7, pp. 349-371, 2003.
- [10] V. Kavitha and M. Punithavalli, "Clustering Time Series Data Stream-A Literature Survey," *Arxiv preprint arXiv:1005.4270*, 2010.
- [11] T. Warren Liao, "Clustering of time series data—a survey," *Pattern Recognition*, vol. 38, pp. 1857-1874, 2005.
- [12] H. Ding, *et al.*, "Querying and mining of time series data: experimental comparison of representations and distance measures," *Proceedings of the VLDB Endowment*, vol. 1, pp. 1542-1552, 2008.
- [13] S. Hirano and S. Tsumoto, "Empirical comparison of clustering methods for long time-series databases," *Active Mining*, pp. 268-286, 2005.
- [14] Z. Zhang, *et al.*, "Comparison of Similarity Measures for Trajectory Clustering in Outdoor Surveillance Scenes," presented at the Proceedings of the 18th International Conference on Pattern Recognition - Volume 03, 2006.
- [15] D. Sankoff and J. Kruskal, "Time warps, string edits, and macromolecules: the theory and practice of sequence comparison," 1983.
- [16] S. Chu, *et al.*, "Iterative deepening dynamic time warping for time series," 2002.
- [17] V. Vuori and J. Laaksonen, "A comparison of techniques for automatic clustering of handwritten characters," *Pattern Recognition*, vol. 3, p. 30168, 2002.
- [18] V. Hautamaki, *et al.*, "Time-series clustering by approximate prototypes," 2008, pp. 1-4.
- [19] B. Morris and M. Trivedi, "Learning trajectory patterns by clustering: Experimental studies and comparative evaluation," 2009, pp. 312-319.
- [20] C. Lai, *et al.*, "A novel two-level clustering method for time series data analysis," *Expert Systems with Applications*, vol. 37, pp. 6319-6326, 2010.
- [21] M. Vlachos, *et al.*, "Discovering Similar Multidimensional Trajectories," presented at the Proceedings of the 18th International Conference on Data Engineering, 2002.
- [22] J. Bezdek, "Fuzzy Mathematics in Pattern Classification," Cornell University, Ithaca, 1973.
- [23] E. Cox, *Fuzzy Modeling and Genetic Algorithms for Data Mining and Exploration*, 2008.
- [24] M. Halkidi, *et al.*, "Clustering algorithms and validity measures," 2002, pp. 3-22.

